اصول پردازش تصویر

نیم‌سال اول ۱۳۹۹-۱۴۰۰

مدرس: دکتر مصطفی کمالی تبریزی

دانشکده علوم ریاضی

تمرین سری سوم ــ سوال پنجم

نام و نام‌خانوادگی: سیدعلیرضا خادم

شماره دانشجویی: ۹۷۱۰۰۳۹۸

موارد لازم.

برای اجرا لازم است تا تصویر tasbih.jpg در مسیر /EX3_Q5/images قرار داشته باشد. همچنین در پیاده‌سازی این سوال از کتابخانه‌های numpy ، skimage و cv2 استفاده شده است که قبل از اجرا بایستی این کتابخانه‌ها روی سیستم شما نصب باشد.

روند کلی حل.

کلیات روش active contourای که پیاده‌سازی شده است مشابه روشی است که در کلاس توضیح داده شده اما یه سری تفاوت‌هایی نیز دارد. که این تفاوت ها عبارت اند از : یکی از تفاوت‌های این پیاده سازی در این است که تعداد راس‌هایی که کانتور را شکل می‌دهند ثابت نیستند و در صورتی که فاصله بین دو راس از یک حدی بیشتر شود یک راس بین این دو راس اضافه می‌شود و همچنین اگر فاصله بین دو راس از حدی کمتر شود یکی از آن رئوس حذف می‌شود. تفاوت دیگر این پیاده سازی این است که در این پیاده سازی علاوه بر انرژی اینترنال و اکسترنال یک انرژی دیگر تحت عنوان انرژی مرکز اضافه شده است که باعث می‌شود که کانتور بتواند وارد قسمت‌های مقعر شکل نیز بشود. انرژی مرکز به صورت میانگین توان ۲ فاصله‌های رئوس از نقاطی که کاربر در ابتدا به عنوان مرکز مشخص می‌کند تعریف می‌شود. (در ابتدای اجرای برنامه که تصویر برای کاربرنمایش داده می‌شود تا منحنی اولیه را مشخص کند، کاربر با کلیک چپ موس می‌تواند رئوس کانتور و با کلیک راست می‌تواند مرکزها را مشخص کند. در صورتی که مرکزی توسط کاربر مشخص نشود، مرکز خود کانتور به عنوان مرکز در نظر گرفته می‌شود، اما توصیه می‌شود با توجه به اینکه آپدیت کردن مرکز کانتور زمان بر است، کاربر خود یک مرکز مانند شکل زیر مشخص کند. البته می‌توانیم متناسب با سوال چند مرکز هم داشته باشیم.)

با توجه به اینکه تعداد راس ها متغیر است، متوقف شدن را بر اساس حداکثر تعداد iteration تعیین می‌کنیم.

توضیح کد.

برنامه در مجموع حاوی ۲ فایل با فرمت py. می‌باشد که توضیحات هر فایل در پایین آمده است.

○ utilities.py

add_remove_vertex(vertices, add_threshold, remove_threshold)

این تابع مجموعه راس‌ها را به همراه دو آستانه برای add و remove کردن می‌گیرد و در صورتی که فاصله دو راس از آستانه add بیشتر باشد بین آن‌ها یک راس اضافه می‌کند و اگر فاصله دو راس از آستانه remove کمتر باشد یکی از دو راس را حذف می‌کند.

cal_internal_energy(vertices, index)

این تابع مجموعه رئوس و یک اندیس را به عنوان ورودی می‌گیرد و میزان تاثیر راسی که اندیس به آن اشاره می‌کند و راس بعدی آن را در انرژی درونی محاسبه کرده و به عنوان خروجی برمی‌گرداند.

cal_average_length_of_contoure(vertices)

این تابع یک تصویر تابع مجموعه رئوس کانتور را به عنوان ورودی می‌گیرد و میانگین فاصله هر دو راس متوالی را به عنوان خروجی برمی‌گرداند.

cal_external_energy(edge_detected_image, vertices, index)

این تابع مجموعه رئوس و یک اندیس و تصویر مرزها را به عنوان ورودی می‌گیرد و میزان تاثیر راسی که اندیس به آن اشاره می‌کند و را در انرژی بیرونی محاسبه کرده و به عنوان خروجی برمی‌گرداند.

cal_centers_energy(vertices, index, centers)

این تابع مجموعه رئوس و یک اندیس و مرکزهارا به عنوان ورودی می‌گیرد و در صورتی که کابر نقطه یا نقاطی را به عنوان مرکز داده باشد مجموع توان ۲ فاصله راسی که اندیس به آن اشاره می‌کند، با مرکزها را به عنوان خروجی برمی‌گرداند. در صورتی کاربر هیچ نقطه‌ای را به عنوان مرکز مشخص نکرده باشد توان ۲ فاصله راسی که اندیس به آن اشاره می‌کند، با مرکز کانتور را به عنوان خروجی برمی‌گرداند.

cal_energy(...)

این تابع یه مجموعه راس و تصویر مرز ضرایب انرژی‌ها، میانگین فاصله دو راس مجاور و مرکزها را به عنوان ورودی می‌گیرد و سه انرژی درونی ، بیرونی و مرکز را محاسبه کرده و به عنوان خروجی برمی‌گرداند.

apply_sobel_filter_x(gray_image)

این تابع یک تصویر سیاه سفید را به عنوان ورودی می‌گیرد و فیلتر sobel_x را برای پیدا کردن مرزهای عمودی روی تصویر اعمال می‌کند و نتیجه را به توان ۲ رسانده و به عنوان خروجی برمی‌گرداند.

apply_sobel_filter_y(gray_image)

این تابع یک تصویر سیاه سفید را به عنوان ورودی می‌گیرد و فیلتر sobel_y را برای پیدا کردن مرزهای افقی روی تصویر اعمال می‌کند و نتیجه را به توان ۲ رسانده و به عنوان خروجی برمی‌گرداند.

get_gradient(src_image)

این تابع یک تصویر را به عنوان ورودی می‌گیرد و بعد از آنکه آن را به تصویر سیاه‌سفید تبدیل با استفاده از توابع apply_sobel_filter_x(gray_image) و apply_sobel_filter_y(gray_image) مرزهای تصویر سیاه‌سفید شده را به دست می‌آورد و به عنوان خروجی برمی‌گرداند.

iteration()

این تابع پیاده سازی contour active با استفاده از برنامه‌نویسی پویا است. تمامی توابعی که در این تابع به کار گرفته شده است توضیح داده شده.

حلقه while تا زمانی که تعداد iteration ها از max_itration کمتر باشد اجرا می‌شود و در هر بار اجرای این حلقه داریم:

در ابتدا میانگین توان ۲ فاصله بین هر دو راس مجاور با استفاده از تابع cal_average_lenght_of_contour محاسبه می‌شود، ماتریس data با ابعاد $(k\_window * k\_window, len(vertices), 2)$ تشکیل می‌دهیم. ماتریس data همان ماتریسی است که برای حل پیدا کردن بهترین جابه‌جایی در روش برنامه‌ریزی پویا استفاده می‌شود. $data[i, k, 0]$ نشان‌دهنده این است که اگر راس k از کانتور را در نظر بگیریم و در همسایگی i از آن قرار داشته باشیم بهترین ردیف که می‌توان از ستون k ـ ۱ انتخاب کرد تا از آن به همسایگی i از راس k برسیم و انرژی کمینه بشود. $data[i, k, 1]$

نشان‌دهنده این است که اگر راس k از کانتور را در نظر بگیریم و در همسایگی i از آن قرار داشته باشیم مجموع انرژی بهترین مسیر تا به اینجا چقدر بوده.

در ادامه جزئیات پیاده‌سازی برنامه‌نویسی پویا آمده است که مطابق با روشی است که در کلاس مطرح شده است.

- **q۵.py**

در این فایل ابتدا تصویر tasbih.jpg را از مسیر /EX3_Q5/images لود می‌کنیم، بعد یک فیلتر bilateral روی تصویر اعمال می‌کنیم تا نویزهای تصویر با حفظ مرزها کمتر بشوند. بعد با استفاده از تابع get_gradient که تصویر سیاه‌سفید شده را به عنوان ورودی می‌گیرد، مرزهای آن را به‌دست می‌آوریم. در ادامه در پنجره‌ای که برای کاربر نمایش داده می‌شود با کلیک چپ روی تصویر می‌توانیم تعدادی نقطه را برای منحنی اولیه تسبیح انتخاب کنیم. بعد با استفاده از کلیک راست مرکز تسبیح را مانند شکل زیر انتخاب می‌کنیم. بعد از انتخاب کاربر می‌بایست دکمه Esc را بفشارد تا برنامه ادامه پیدا کند.

در نهایت تابع iteration را فراخوانی می‌کنیم تا کانتور حرکت کرده و به مرزهای تسبیح بچشبد.